Multi-Item Multiperiodic Inventory Control Problem with Variable Demand and Discounts: A Particle Swarm Optimization Algorithm

Seyed Mohsen Mousavi, S. T. A. Niaki, Ardeshr Bahreininejad, and Siti Nurmaya Musa

Department of Mechanical Engineering, Faculty of Engineering, University of Malaya, 50603 Kuala Lumpur, Malaysia
Department of Industrial Engineering, Sharif University of Technology, P.O. Box 11155-0414 Azadi Avenue, Tehran 1458889684, Iran

Correspondence should be addressed to Ardeshr Bahreininejad; bahreininejad@um.edu.my

Received 21 April 2014; Accepted 28 May 2014; Published 30 June 2014

Copyright © 2014 Seyed Mohsen Mousavi et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

A multi-item multiperiod inventory control model is developed for known-deterministic variable demands under limited available budget. Assuming the order quantity is more than the shortage quantity in each period, the shortage in combination of backorder and lost sale is considered. The orders are placed in batch sizes and the decision variables are assumed integer. Moreover, all unit discounts for a number of products and incremental quantity discount for some other items are considered. While the objectives are to minimize both the total inventory cost and the required storage space, the model is formulated into a fuzzy multicriteria decision making (FMCDM) framework and is shown to be a mixed integer nonlinear programming type. In order to solve the model, a multiobjective particle swarm optimization (MOPSO) approach is applied. A set of compromise solution including optimum and near optimum ones via MOPSO has been derived for some numerical illustration, where the results are compared with those obtained using a weighting approach. To assess the efficiency of the proposed MOPSO, the model is solved using multi-objective genetic algorithm (MOGA) as well. A large number of numerical examples are generated at the end, where graphical and statistical approaches show more efficiency of MOPSO compared with MOGA.

1. Introduction and Literature Review

Most real-world problems in industries and commerce are studied as an optimization problem involving a single objective. The assumption that organizations always seek to maximize (or minimize) their profit (or their cost) rather than making trade-offs among multiple objectives has been censured for a long time. Generally, classical inventory models are developed under the basic assumption that the management purchases or produces a single product. However, in many real-life conditions, this assumption does not hold. Instead, many firms, enterprises, or vendors are motivated to store a number of products in their shops for more profitable business affairs. Another cause of their motivation is to attract the customers to purchase several items in one showroom or shop.

This work proposes a multiperiod inventory model for seasonal and fashion items. The multiperiodic inventory control problems have been investigated in depth in different research. Chiang [1] investigated a periodic review inventory model in which the period is partly long. The important aspect of his study was to introduce emergency orders to prevent shortages. He employed a dynamic programming approach to model the problem. Mohebbi and Posner [2] investigated an inventory system with periodic review, multiple replenishment, and multilevel delivery. Assuming a Poisson process for the demand, shortages were allowed in this research, and the lost sale policy could be employed. Lee and Kang [3] developed a model for managing inventory of a product in multiple periods. Their model was first derived for one item and then was extended for several products. Mousavi et al. [4] proposed a multiproduct multiperiod inventory control problem under time value of money and inflation where total storage space and budget were limited. They solved the problem using two metaheuristic algorithms,
that is, genetic algorithm and simulated annealing. Mirzapour Al-e-hashem and Rekik [5] presented a multiproduct multi-period inventory routing problem, where multiple constrained vehicles distributed products from multiple suppliers to a single plant to meet the given demand of each product over a finite planning horizon. Janakiraman et al. [6] analyzed the multiperiodic news vendor problem and proposed some new results.

The quantity discount is of increasing attention due to its practical importance in purchasing and control of items. Usually, one derives the better marginal cost of purchase/production by taking advantages of the chances of cost savings through bulk purchase/production. Furthermore, in supply chain environments, quantity discounts can be considered an inventory coordination mechanism between buyers and suppliers [7]. In the literature of quantity discounts, Benton [8] considered an inventory system with quantity discount with multiple price breaks and alternative purchasing and lot-sizing policy. Abad [9, 10] proposed models for joint price and lot size determination when supplier offers either incremental (IQD) or all unit (AUD) quantity discounts. K. Maiti and M. Maiti [11] developed a model for a multi-item inventory control system of breakable items with AUD and IQD (and a combination of the two policies) and proposed genetic algorithm to solve the model. Sana and Chaudhuri [12] extended an EOQ model by relaxation of the preassumptions related to payments, allowing delay on delivery and discounts. They used a mixed integer nonlinear programming technique to model the problem. Taleizadeh et al. [13] considered a genetic algorithm to optimize multiproduct multi-item inventory control systems with stochastic replenishment intervals and discount. Recently, several works such as the ones in [4, 14–16] have also spotted discounts in inventory control problems. Huang and Lin [17] addressed an integrated model that scheduled multi-item replenishment with uncertain demand to determine delivery routes and truck loads. In this study, the products are purchased in different periods under AUD and IQD policies.

Metaheuristic algorithms have been suggested to solve some of the existing developed inventory problems in the literature. Some of these algorithms are tabu search [18, 19, 31], genetic algorithms (GA) [20–22, 32], simulating annealing (SA) [23, 33, 34], evolutionary algorithm [24, 35], threshold accepting [30], neural networks [36], ant colony optimization [37], fuzzy simulation [25], and harmony search [26, 38–41].

Inspired by social behavior of bird flocking or fish schooling, particle swarm optimization (PSO) is also a population-based stochastic optimization metaheuristic developed by Kennedy and Eberhart [42]. Recently, researchers have employed this effective technique to find optimal or near optimal solutions of their inventory control problems. For example, Taleizadeh et al. [43] employed PSO to solve their integer nonlinear programming model of a constraint joint single buyer-single vendor inventory problem with changeable lead time and \((r, Q)\) policy in supply chains with stochastic demand. Chen and Dye [44] solved an inventory problem with deteriorating products and variable demands using a PSO algorithm. Further, Taleizadeh et al. [27] modeled a chance-constraint supply chain problem with uniformly distributed stochastic demand, where an Ant Colony Bee and a PSO algorithm were utilized to solve the problem.

Instead of optimizing a single objective, some researchers tried to find Pareto front solutions for their multiple objective inventory planning problems that usually consist of multiple conflicting objectives. Agheli [45] proposed a multicriteria framework for inventory control problem, in which the solution procedure was an interactive method with preferences extracted gradually in decision analysis process to determine batch size and security stock. Roy and Maiti [28] presented a multiobjective inventory model of deteriorating items with stock-dependent demand under limited imprecise storage area and total cost budget. Tsou [46] developed a multiobjective reorder point and order size system and proposed a multiobjective PSO (MOPSO) to generate Pareto front solutions. He employed TOPSIS to sort the nondominated solutions. The objectives therein were to maximize the profit and to minimize the wastage cost where the profit goal, wastage cost, and storage area were fuzzy in nature. One of the successful applications of PSO to MOOPs is the seminal work of Coello Coello and Lechuga [47]. Yaghin et al. [29] first addressed an inventory-marketing system to determine the production lot size, marketing expenditure, and selling prices in which the model was formulated as a fuzzy nonlinear multiobjective program. Then, they converted the model to a classical single-objective one by a fuzzy goal programming method where an efficient solution procedure using PSO was provided to solve the resulting nonlinear problem. In their study, MOPSO is not only a viable alternative to solve MOOPs, but also the only one, compared with the nondominated sorting genetic algorithm-II (NSGA-II) [48], the Pareto archive evolutionary strategy (PAES) [49], and the microgenetic algorithm [50] for multiobjective optimization problems [51]. Table 1 shows the literature review of the works reviewed in this work where DOE is an abbreviation of term “design of experiments.”

In this research, the contribution of the problem is considering a new biobjective multi-item multiperiodic inventory control model where some items are purchased under AUD and the other items are bought from IQD. The demands vary in different periods, the budget is limited, the orders are placed in batch sizes, and shortages in combination of backorder and lost sale are considered. The goal is to find the optimum inventory levels of the items in each period such that the total inventory cost and the total required warehouse space are minimized simultaneously. Since it is not easy for the managers to allocate the crisp values to the weights of the objectives in a decision making process, considering these weights as fuzzy numbers will be taken as an advantage.

In order to be more understanding of the problem, we try to explain the model with taking an example in the real world. We consider a company which produces some kinds of fashion clothes including trousers, t-shirt, and shirt in a certain period. The customers (wholesales) of this company with different demand rates make the orders and receive their products in the prespecific boxes, each one consisting of a known number of these clothes. Moreover, due to some unforeseen matters, such as production limitation, the
companies are not responsive to all of the demands in a period and hence some customers must wait until the next period to receive their orders. Furthermore, it is assumed the company is going to extend the production part and therefore the owner has a plan to build and optimize a new storage subject to the available space.

The remainder of the paper is organized as follows. In Section 2, the problem along with its assumptions is defined. In Section 3, the defined problem of Section 2 is modeled. To do this, the parameters and the variables of the problem are first introduced. A MOPSO algorithm is presented in Section 4 to solve the model. Section 5 contains a numerical example for a problem with 5 items and 3 periods, for which a multiobjective genetic algorithm (MOGA) is also applied as benchmark for comparisons. Finally, conclusion and recommendations for future research come in Section 6.

2. Problem Definition, Assumptions, and Notations

Consider a biobjective multi-item multiperiod inventory control problem, in which an AUD policy is used for some items and an IQD policy for some other items. The inventory control problem of this research is similar to the seasonal items problem where the planning horizon starts in a period (or season) and finish in a certain period (or season). The total available budget in the planning horizon is limited and fixed. Due to existing ordering limitations or production constraints, the order quantities of all items in different periods cannot be more than their predetermined upper bounds. The demands of the products are constant and distinct, and, in case of shortage, a fraction is considered backorder and a fraction lost sale. The costs associated with the inventory control system are holding, backorder, lost sale, and purchasing costs. Moreover, due to current managerial decision adaptations on production policies (i.e., setting up a new manufacturing line, extending the warehouse, or building a new storage area), minimizing the total storage space is required as well as minimizing the total inventory costs. Therefore, the goal is to identify the inventory levels of the items in each period such that the two objective functions, total inventory costs and total storage space, are minimized.

In order to simplify the modeling, the following assumptions are set to the problem at hand.

1. The demand rate of an item is independent of the others and is constant in a period. However, it can be different in different periods.

<table>
<thead>
<tr>
<th>References</th>
<th>Multiproduct</th>
<th>Multiperiod</th>
<th>Fuzzy multiobjective</th>
<th>Discount policy</th>
<th>Solving methodology</th>
<th>Shortages</th>
<th>DOE</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1]</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td>B and B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[2]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Level-crossing</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>[4]</td>
<td>✓</td>
<td></td>
<td></td>
<td>IQD</td>
<td>GA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[5]</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>CPLEX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[6]</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>[7]</td>
<td></td>
<td></td>
<td></td>
<td>IQD and AUD</td>
<td>Simulation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[8]</td>
<td></td>
<td></td>
<td></td>
<td>AUD</td>
<td>Simulation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[9]</td>
<td></td>
<td></td>
<td></td>
<td>IQD</td>
<td>Numerical methods</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[10]</td>
<td></td>
<td></td>
<td></td>
<td>IQD</td>
<td>Numerical methods</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[12]</td>
<td>✓</td>
<td></td>
<td></td>
<td>AUD</td>
<td>Numerical methods</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[13]</td>
<td>✓</td>
<td></td>
<td></td>
<td>IQD and AUD</td>
<td>GA</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>[14]</td>
<td></td>
<td></td>
<td></td>
<td>IQD</td>
<td>Yager ranking</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[15]</td>
<td></td>
<td></td>
<td></td>
<td>IQD</td>
<td>Excel macro</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[16]</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td>TOPSIS and GA</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>[17]</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>ACO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[18]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Tabu search and Lagrangian</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>[19]</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td>Tabu search</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[20]</td>
<td></td>
<td></td>
<td></td>
<td>IQD</td>
<td>Goal programming and GA</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>[21]</td>
<td>✓</td>
<td></td>
<td></td>
<td>IQD</td>
<td>GA and fuzzy simulation</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>[22]</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>GA</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>[23]</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>SA and GA</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>[24]</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td>TOPSIS and GA</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>[25]</td>
<td>✓</td>
<td></td>
<td></td>
<td>IQD</td>
<td>fuzzy simulation</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>[26]</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>Harmony search</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>[27]</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>Bee colony and PSO</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>[28]</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>Fuzzy programming algorithm</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>[29]</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td>Fuzzy method</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This research       ✓       ✓       ✓       ✓       ✓       PSO and GA       ✓       Taguchi
(2) At most one order can be placed in a period. This order can include or exclude an item.
(3) The items are delivered in a special container. Thus, the order quantities must be a multiple of a fixed-sized batch.
(4) The vendor uses an AUD policy for some items and an IQD policy for others.
(5) A fraction of the shortages is considered backorder and a fraction lost sale.
(6) The initial inventory level of all items is zero.
(7) The budget is limited.
(8) The planning horizon is finite and known. In the planning horizon, there are \( N \) periods of equal duration.
(9) The order quantity on an item in a period is greater than or equal to its shortage quantity in the previous period (i.e., \( Q_{i,j+1} \geq b_{i,j} \) defined below).

In order to model the problem at hand, in what comes next we first define the variables and the parameters. Then, the problem is formulated in Section 3.

For \( i = 1,2,\ldots,m \) and \( j = 1,2,\ldots,N-1 \) and \( k = 1,2,\ldots,K \) the variables and the parameters of the model are defined as follows:

- \( N \): number of replenishment cycles during the planning horizon,
- \( m \): number of items,
- \( K \): number of price break points,
- \( S_i \): required storage space per unit of the \( i \)th product,
- \( T_j \): total time elapsed up to and including the \( j \)th replenishment cycle,
- \( T_{i,j} \): \( j \)th period in which the inventory level of item \( i \) is zero (a decision variable),
- \( B_i \): batch size of the \( i \)th product,
- \( V_{i,j} \): number of the packets for the \( i \)th product order in period \( j \) (a decision variable),
- \( D_{i,j} \): demand of the \( i \)th product in period \( j \),
- \( Q_{i,j} \): purchase quantity of item \( i \) in period \( j \) (a decision variable where \( Q_{i,j} = B_i V_{i,j} \)),
- \( A_i \): ordering cost per replenishment of product \( i \) (If an order is placed for one or more items in period \( j \), this cost appears in that period),
- \( b_{i,j} \): shortage quantity of the \( i \)th product in period \( j \) (a decision variable),
- \( X_{i,j} \): the beginning positive inventory level of the \( i \)th product in period \( j \) (in \( j = 1 \), the beginning positive inventory level of all items is zero) (a decision variable),
- \( I_{i,j} \): inventory position of the \( i \)th product in period \( j \) (it is \( X_{i,j+1} + Q_{i,j+1}, \) if \( I_{i,j} \geq 0 \), otherwise equals \( b_{i,j} \)),
- \( I_i(t) \): the inventory level of the \( i \)th item at time \( t \),
- \( H_i \): unit inventory holding cost for item \( i \),
- \( q_{i,k} \): \( k \)th discount point for the \( i \)th product (\( q_{i,1} = 0 \)),
- \( m_{i,k} \): discount rate of item \( i \) in \( k \)th price break point (\( m_{i,1} = 0 \)),
- \( P_i \): purchasing cost per unit of the \( i \)th product,
- \( P_{i,k} \): purchasing cost per unit of the \( i \)th product at the \( k \)th price break point,
- \( U_{i,j,k} \): a binary variable, set 1 if item \( i \) is purchased at price break point \( k \) in period \( j \), and 0 otherwise,
- \( W_{i,j} \): a binary variable, set 1 if a purchase of item \( i \) is made in period \( j \), and 0 otherwise,
- \( L_{i,j} \): a binary variable, set 1 if a shortage for item \( i \) occurs in period \( j \), and 0 otherwise,
- \( \beta \): percentage of unsatisfied demands of the \( i \)th product, that is, back ordered,
- \( \pi_{i,j} \): backorder cost per unit demand of the \( i \)th product in period \( j \),
- \( \pi_{i,1} \): shortage cost per unit of the \( i \)th product in period \( j \), that is, lost,
- \( Z_1 \): total inventory cost,
- \( Z_2 \): total storage space,
- \( TB \): total available budget,
- \( M_1 \): an upper bound for order quantity of the \( i \)th item in period \( j \),
- \( M_2 \): an upper bound for order quantities of all items in each period (the truck capacity),
- \( TMF \): objective function (the weighted combination of the total inventory cost and the total storage space),
- \( \omega_1 \): a weight associated with the total inventory cost (\( 0 \leq \omega_1 \leq 1 \)),
- \( \omega_2 \): a weight associated with the total storage space (\( 0 \leq \omega_2 \leq 1 \)).

3. Problem Formulation

A graphical representation of the inventory control problem at hand with 5 periods for item \( i \) is given in Figure 1 to obtain the inventory costs. At the beginning of the primary period \( (T_0) \), it is assumed the starting inventory level of item \( i \) is zero and that the order quantity has been received and is available. In the following periods, shortages can either occur or not. If shortage occurs, the corresponding binary variable \( x \) is 1, otherwise it is zero. In the latter case, the inventory levels at the beginning of each period may be positive.

3.1. The Objective Functions. The first objective function of the problem, the total inventory cost, is obtained as

\[
Z_1 = \text{Total Inventory Cost} = \text{Total Ordering Cost} + \text{Total Holding Cost} \quad (1)
+ \text{Total Shortage Cost} + \text{Total Purchasing Cost},
\]

where each part is derived as follows.
The ordering cost of an item in a period occurs when an order is placed for it in that period. Using a binary variable $W_{i,j}$, where it is 1 if an order for the $i$th product in period $j$ is placed and zero otherwise, and knowing that orders can be placed in periods 1 to $N - 1$ the total ordering cost is obtained as

$$\text{Total Ordering Cost} = \sum_{i=1}^{m} \sum_{j=1}^{N-1} A_i W_{i,j} \tag{2}$$

Since it is assumed a shortage may occur for a product in a period or not, the holding cost derivation is not as straightforward as the ordering cost derivation. Taking advantage of a binary variable $L_{i,j}$, where it is 1 if a shortage for item $i$ in period $j$ occurs and otherwise zero, and using Figure 1, the holding cost for item $i$ in the time interval $T_{j-1} \leq t \leq T_j (1 - L_{i,j}) + T_{i,j}' L_{i,j}$ is obtained as

$$H_i \int_{T_{j-1}}^{T_j(1-L_{i,j})+T_{i,j}' L_{i,j}} I_i(t) \, dt, \tag{3}$$

where $I_i(t)$ is the inventory level of the $i$th item at time $t$. In (3), if a shortage for item $i$ occurs, $L_{i,j}$ becomes 1 and the term $T_j (1 - L_{i,j}) + T_{i,j}' L_{i,j}$ becomes $T_{i,j}'$. Otherwise, $L_{i,j} = 0$ and $T_j (1 - L_{i,j}) + T_{i,j}' L_{i,j} = T_j$. In Figure 1, the trapezoidal area above the horizontal timeline in each period when multiplied by the unit inventory holding cost of an item, $H_i$, represents the holding cost of the item in that period. In other word, since

$$I_{i,j+1} = I_{i,j} + Q_{i,j} - D_{i,j} \tag{4}$$

and if $I_{i,j+1} \geq 0$ then $I_{i,j+1} = X_{i,j+1}$, otherwise $I_{i,j+1} = b_{i,j}$, (3) becomes

$$H_i \int_{T_{j-1}}^{T_j (1-L_{i,j})+T_{i,j}' L_{i,j}} I_i(t) \, dt = \frac{X_{i,j} + Q_{i,j} - D_{i,j}}{2} (T_j (1 - L_{i,j}) + T_{i,j}' L_{i,j} - T_{j-1}) H_i. \tag{5}$$

Therefore, the total holding cost is obtained in

$$\text{Total Holding Cost} = \sum_{i=1}^{m} \sum_{j=1}^{N-1} \left( \frac{X_{i,j} + Q_{i,j} + X_{i,j+1}}{2} \right) \times (T_j (1 - L_{i,j}) + T_{i,j}' L_{i,j} - T_{j-1}) H_i. \tag{6}$$

The total shortage cost consists of two parts: the total backorder cost and the total lost sale cost. In Figure 1, the trapezoidal area underneath the horizontal timeline in each period (shown for the primary period) when multiplied by the backorder cost per unit demand of the $i$th product in period $j$, $\pi_{i,j}$, is equal to the backorder cost of the item in that period. Therefore, the total backorder cost will be

$$\text{Total Backorder Cost} = \sum_{i=1}^{m} \sum_{j=1}^{N-1} \left( \frac{\pi_{i,j} b_{i,j}}{2} (T_{j} - T_{i,j}') \right) \beta_i \tag{7}.$$

Furthermore, since $(1 - \beta_i)$ represents the percentage demands of the $i$th product, that is, lost sale, the total lost sale becomes

$$\text{Total Lost Sale Cost} = \sum_{i=1}^{m} \sum_{j=1}^{N-1} \left( \frac{\bar{\pi}_{i,j} b_{i,j}}{2} (T_{j} - T_{i,j}') (1 - \beta_i) \right) \tag{8}$$

in which $T_{j} - T_{i,j}' = b_{i,j} / D_{i,j}$.

The total purchase cost also consists of two AUD and IQD costs. The purchasing offered by AUD policy is modeled by

$$P_i = \begin{cases} P_{i,1}: & 0 < Q_{i,1} \leq q_{i,2} \\ P_{i,2}: & q_{i,2} < Q_{i,1} \leq q_{i,3} \\ \vdots \\ P_{i,K}: & q_{i,K} < Q_{i,1} \end{cases} \tag{9}$$

Hence, the purchasing cost of this policy is obtained as

$$\text{AUD Purchasing Cost} = \sum_{i=1}^{m} \sum_{j=1}^{N-1} \sum_{k=1}^{K} P_{i,k} Q_{i,j} U_{i,j,k}. \tag{10}$$

Figure 1: Some possible situations for the inventory of item $i$ in 5 periods.
A graphical representation of the AUD policy employed to purchase the products in different periods is shown in Figure 2. In this Figure, the relation between the price break points and the purchasing costs is demonstrated clearly. Moreover, $U_{i,j,k}$ is a binary variable, set 1 if the $i$th item is purchased with price break $k$ in period $j$ and 0 otherwise.

In the IQD policy, the purchasing cost per unit of the $i$th product depends on its order quantity. Therefore, for each price break point we have

$$
\begin{align*}
    P_{i,1}q_{i,2} + P_{i,2}(q_{i,3} - q_{i,2})(1 - m_{i,2}) + \cdots + P_{i,K}(Q_{i,j} - q_{i,K})(1 - m_{i,K})
\end{align*}
$$

Hence, the total purchasing cost under the IQD policy is obtained as

IQD Purchasing Cost

$$
= \sum_{i=1}^{m} \sum_{j=1}^{N-1} \left\{ (Q_{i,j} - q_{i,k}) P_{i,j,k} (1 - m_{i,k}) + \sum_{k=1}^{K-1} (q_{i,k+1} - q_{i,k}) P_{i,k+1} (1 - m_{i,k}) \right\}.
$$

Figure 3 graphically depicts the IQD policy for each product in different periods.

Thus, the first objective function of the problem at hand becomes

$$
Z_1 = \sum_{i=1}^{m} \sum_{j=1}^{N-1} \left( X_i + \sum_{j=1}^{N-1} (Q_{i,j} + X_{i,j+1}) \right) + \sum_{i=1}^{m} \sum_{j=1}^{N-1} \left( \mathbf{1}_{k} (Q_{i,j} - q_{i,K}) U_{i,j,k} (1 - m_{i,K}) \right) + \sum_{i=1}^{m} \sum_{j=1}^{N-1} \left( \mathbf{1}_{k} (Q_{i,j} - q_{i,k}) P_{i,k+1} (1 - m_{i,k}) \right).\tag{13}
$$

The second objective of the problem is to minimize the total required storage space. Since in each period, order quantities $Q_{i,j}$ enter the storage and the beginning inventory
the second objective function of the problem is modeled by

\[ Z_2 = \sum_{i=1}^{m} \sum_{j=1}^{N-1} \left( X_{i,j} + Q_{i,j} \right) S_i. \]  \tag{14}

Finally, the fitness function is defined as the weighted combination of the total inventory cost and the required storage space as

\[ TMF = w_1 Z_1 + w_2 Z_2. \]  \tag{15}

3.2. The Constraints. In real-world inventory planning problems, due to existing constraints on either supplying or producing goods (e.g., budget, labor, production, carrying equipment, and the like), objectives are not met simply. This section presents formulations for some real-world constraints.

The first limitation is given in (4), where it relates the beginning inventory of the items in a period to the beginning inventory of the items in the previous period plus the order quantity of the previous period minus the demand of the previous period.

The second limitation is due to delivering the items in packets of batches. Since \( Q_{i,j} \) represents the purchase quantity of item \( i \) in period \( j \), denoting the batch size by \( B_i \) and the number of packets by \( V_{i,j} \), we have

\[ Q_{i,j} = B_i V_{i,j}. \]  \tag{16}

Furthermore, since \( Q_{i,j} \) can only be purchased based on one price break point, the following constraint must hold:

\[ \sum_{k=1}^{K} U_{i,j,k} = 1. \]  \tag{17}

The prerequisite of using this strategy is that the lowest \( q_{i,k} \) in the AUD table must be zero (i.e., \( q_{i,1} = 0 \)).

Since the total available budget is \( TB \), the unit purchasing cost of the product is \( P_i \), and the order quantity is \( Q_{i,j} \), the budget constraint will be

\[ \sum_{i=1}^{m} \sum_{j=1}^{N-1} Q_{i,j} P_i \leq TB. \]  \tag{18}

In real-world environments, the order quantity of an item in a period may be limited. Defining \( M_i \) an upper bound for this quantity, for \( i = 1, 2, \ldots, m \) and \( j = 1, 2, \ldots, N - 1 \) we have

\[ Q_{i,j} \leq M_i. \]  \tag{19}

Moreover, due to transportation contract and the truck capacity, the number of product orders and the total order quantities in a period are limited as well. Hence, for \( j = 1, 2, \ldots, N - 1 \), we have

\[ \sum_{i=1}^{m} Q_{i,j} W_{i,j} \leq M_2. \]  \tag{20}

where if an order occurs for item \( i \) in period \( j \), \( W_{i,j} = 1 \), otherwise \( W_{i,j} = 0 \). Further, \( M_j \) is an upper bound on the total number of orders and the total order quantities in a period.

As a result, the complete mathematical model of the problem is

\[ \text{Min } \text{TMF} = w_1 Z_1 + w_2 Z_2 \]  \tag{21}

subject to

\[ Z_1 = \sum_{i=1}^{m} \sum_{j=1}^{N-1} A_i W_{i,j} + \sum_{i=1}^{m} \sum_{j=1}^{N-1} \left( \frac{X_{i,j} + Q_{i,j} + X_{i,j+1}}{2} \right) \times \left( T_j \left( 1 - L_{i,j} \right) + T_{i,j} L_{i,j} - T_{i,j-1} \right) H_i \]

\[ + \sum_{i=1}^{m} \sum_{j=1}^{N-1} \left( \frac{\pi_{i,j} b_{i,j}}{2} \left( T_j - T_{i,j} \right) \beta_i \right) \]

\[ + \sum_{i=1}^{m} \sum_{j=1}^{N-1} \sum_{k=1}^{K} Q_{i,j} P_{i,k} U_{i,j,k} \]

\[ + \sum_{i=1}^{m} \sum_{j=1}^{N-1} \left\{ \left( q_{i,j} - q_{i,k} \right) P_{i,j,k} (1 - m_{i,k}) \right\} + \sum_{k=1}^{K-1} \left( q_{i,j,k+1} - q_{i,j,k} \right) P_{i,j,k+1} (1 - m_{i,k+1}) \}

\[ Z_2 = \sum_{i=1}^{m} \sum_{j=1}^{N-1} \left( X_{i,j} + Q_{i,j} \right) S_i. \]

\[ L_{i,j+1} = \begin{cases} I_{i,j+1} + Q_{i,j} - D_{i,j}, & \text{if } i = 1, 2, \ldots, m, \ j = 1, 2, \ldots, N - 1, \\ 0, & \text{otherwise} \end{cases} \]

\[ I_{i,j+1} = \begin{cases} X_{i,j+1}, & I_{i,j+1} \geq 0, \\ B_{i,j} - I_{i,j+1}, & I_{i,j+1} < 0, \end{cases} \]

\[ \sum_{k=1}^{K} U_{i,j,k} = 1; \]

\[ Q_{i,j} = B_i V_{i,j}; \]

\[ \sum_{i=1}^{m} \sum_{j=1}^{N-1} Q_{i,j} P_i \leq TB \]

\[ Q_{i,j} \leq M_i; \]
\[(i = 1, 2, \ldots, m), \quad (j = 1, 2, \ldots, N - 1)\]

\[W_{i,j} \in \{0, 1\}; \quad (j = 1, 2, \ldots, N - 1)\]

\[U_{i,j,k} \in \{0, 1\}; \quad (i = 1, 2, \ldots, m), \quad (j = 1, 2, \ldots, N - 1), \quad (k = 1, 2, \ldots, K)\]

\[\sum_{i=1}^{m} Q_{i,j} W_{i,j} \leq M_{j}; \quad (j = 1, 2, \ldots, N - 1)\]

\[Q_{i,j+1} \geq b_{i,j} \quad (22)\]

In most inventory-planning models that have been developed so far, researchers have imposed some unrealistic assumptions such that the objective function of the model becomes concave and the model can easily be solved by some mathematical approaches like the Lagrangian or the derivative methods. However, since the model in (22), which is obtained based on assumptions that are more realistic, is an integer nonlinear programming mixed with binary variables, reaching an analytical solution (if any) to the problem is difficult. In addition, efficient treatment of integer nonlinear optimization is one of the most difficult problems in practical optimization [52]. As a result, in the next section a metaheuristic algorithm is proposed to solve the model in (22).

4. The Proposed Multiobjective Particle Swarm Optimization Algorithm

Many researchers have successfully used metaheuristic methods to solve complicated optimization problems in different fields of scientific and engineering disciplines; among them, the particle swarm optimization (PSO) algorithm is one of the most efficient methods. That is why this approach is taken in this research to solve the model in (22). The structure of the proposed MOPSO that is based on the PSO algorithm for the multiobjective inventory planning problem at hand is given as follows.

4.1. Generating and Initializing the Particles Positions and Velocities. PSO is initialized by a group of random particles (solutions) called generation and then searches for optima by updating generations. The initial population is constructed by randomly generated \( R \) particles (similar to the chromosomes of a genetic algorithm). In a \( d \)-dimensional search space, let \( x^{i}_k = \{x^{i}_{k,1}, x^{i}_{k,2}, \ldots, x^{i}_{k,d}\} \) and \( v^{i}_k = \{v^{i}_{k,1}, v^{i}_{k,2}, \ldots, v^{i}_{k,d}\} \) be, respectively, the position and the velocity of particle \( i \) at time \( k \). Then, (23) is applied to generate initial particles, in which \( x^{\text{min}} \) and \( x^{\text{max}} \) are the lower and the upper bounds on the design variable values and RAND is a random number between 0 and 1. Consider

\[x^{i}_0 = x^{\text{min}} + \text{RAND} (x^{\text{max}} - x^{\text{min}})\]

\[v^{i}_0 = x^{\text{min}} + \text{RAND} (x^{\text{max}} - x^{\text{min}})\]  \hspace{1cm} (23)

4.2. Selecting the Best Position and Velocity. For every particle, denote the best solution (fitness) that has been achieved so far as

\[p^{\text{best}}^{i,k} = \{p^{\text{best}}^{i,k,1}, p^{\text{best}}^{i,k,2}, \ldots, p^{\text{best}}^{i,k,d}\} \quad (24)\]

\[g^{\text{best}}^{i,k} = \{g^{\text{best}}^{i,k,1}, g^{\text{best}}^{i,k,2}, \ldots, g^{\text{best}}^{i,k,d}\} \quad (25)\]

An important note for the generating and initializing step of the PSO is that solutions must be feasible and must satisfy the constraints. As a result, if a solution vector does not satisfy a constraint, the related vector solution will be penalized by a big penalty on its fitness.

4.3. Velocity and Position Update. The new velocities and positions of the particles for the next fitness evaluation are calculated using [53, 54]

\[v^{i,k+1,d} = w \cdot v^{i,k,d} + C_1 \cdot r_1 \cdot \left(\text{p^{best}}^{i,k,d} - x^{i,k,d}\right)\]

\[+ C_2 \cdot r_2 \cdot \left(\text{g^{best}}^{i,k,d} - x^{i,k,d}\right)\]

\[x^{i,k+1,d} = x^{i,k,d} + v^{i,k+1,d}\]  \hspace{1cm} (26)

where \( r_1 \) and \( r_2 \) are random numbers between 0 and 1, coefficients \( C_1 \) and \( C_2 \) are given acceleration constants towards \( \text{p^{ best}} \) and \( \text{g^{ best}} \), respectively, and \( w \) is the inertia weight. Introducing a linearly decreasing inertia weight into the original PSO significantly improves its performance through the parameter study of inertia weight [55, 56]. Moreover, the linear distribution of the inertia weight is expressed as follows [55]:

\[w = w^{\text{max}} - \frac{w^{\text{max}} - w^{\text{min}}}{\text{iter}^{\text{max}}} \cdot \text{iteration}\]  \hspace{1cm} (27)

where \( \text{iter}^{\text{max}} \) is the maximum number of iterations and iteration is the current number of iteration. Equation (27) presents how the inertia weight is updated, considering \( w^{\text{max}} \) and \( w^{\text{min}} \) are the initial and the final weights, respectively. The parameters \( w^{\text{max}} = 0.9 \) and \( w^{\text{min}} = 0.4 \) that were previously investigated by Naka et al. [55] and Shi and Eberhart [56] are used in this research as well.

4.4. Stopping Criterion. Achieving a predetermined solution, steady-state mean, and standard deviations of a solution in several consecutive generations, stopping the algorithm at a certain computer CPU time, or stopping when a maximum number of iterations is reached are usual stopping rules that have been used so far in different research works. In current research, the PSO algorithm stops when the maximum number of iterations is reached.

Pseudocode 1 shows the pseudocode of the proposed MOPSO algorithm. Moreover, since the problem and hence
for $i = 1$ to Pop
  initialize position ($i$)
  initialize velocity ($i$)
  if position ($i$) and velocity ($i$) be a feasible candidate solution
    penalty = 0
  else penalty = a positive number
  endif
end for

$w = [0.4, 0.9]$
do while Iter $\leq$ Gen
  for $j = 1$ to Pop
    Calculate new velocity of the particle
    Calculate new position of the particle
    $p_{\text{best}}(\text{iter}) = \min(p_{\text{best}}(i))$
  end for
  $g_{\text{best}}(\text{iter}) = \min(g_{\text{best}})$
  $w = w_{\text{max}} - \frac{(w_{\text{max}} - w_{\text{min}})}{\text{iter}_{\text{max}}} \times \text{iter}$
  modifying the velocity and position of the particle
end while

Pseudocode 1: The pseudocode of MOPSO algorithm.

Begin
  Set $P_C, P_m$, Pop and Gen
  $S \rightarrow 0$
  initialize Population ($S$)
  evaluate Population ($S$)
  while (non terminating condition)
    repeat
      $S \rightarrow S + 1$
      select Population ($S$) from Population with roulette wheel ($S - 1$)
      uniform crossover
      one point random mutation
      evaluate Population ($S$)
    end repeat
    print optimum result
end

Pseudocode 2: The pseudocode of MOGA algorithm.

the model is new and there is no other available algorithm
to compare the results, a multiobjective genetic algorithm
(MOGA) is developed in this research for validation and
benchmarking. MOGA was coded using roulette wheel in
selection operator, population size of 40, uniform crossover
with probability of 0.64, one-point random mutation with
probability 0.2, and a maximum number of 500 iterations.
Pseudocode 2 shows the pseudocode of the proposed MOGA
algorithm. The computer programs of the MOPSO and
MOGA algorithms were developed in MATLAB software and
are executed on a computer with 2.50 GHz of core 2 CPU
and 3.00 GB of RAM. Furthermore, all the graphical and
statistical analyses are performed in MINITAB 15.

In the next section, some numerical examples are given to
illustrate the application of the proposed MOPSO algorithm
in real-world environments and to evaluate and compare its
performances with the ones obtained by a MOGA method.

5. Numerical Illustrations

The decision variables in the inventory model (22) are $Q_{i,j}$,
$X_{i,j}$, $V_{i,j}$, and $b_{i,j}$. We note that the determination of the order
quantity of the items in different periods, that is, $Q_{i,j}$, results
in the determination of the other decision variables as well.
Hence, we first randomly generate $Q_{i,j}$, that is, modeled by
the particles’ position and velocity. Equation (28) shows a
pictorial representation of the matrix $Q$ for a problem with
4 items in 4 periods, where rows and columns correspond to
the items and the periods, respectively.

The structure of a particle

$$Q_{4,4} = \begin{bmatrix}
124 & 116 & 50 & 0 \\
205 & 190 & 58 & 0 \\
114 & 68 & 107 & 0 \\
43 & 87 & 210 & 0
\end{bmatrix}.$$

(28)
Table 2 shows partial data for 40 different problems with different sizes along with their near optimal solutions obtained by MOPSO and MOGA. In these problems, the number of items varies between 1 and 20 and the number of periods takes values between 3 and 15. In addition, the total available budgets and the upper bounds for the order quantities \( M_1 \) are given in Table 1 for each problem.

In order to illustrate how the results are obtained, consider a typical problem with 5 items and 3 periods (the seventh row in Table 2), for which the complete input data is given in Table 3. The parameters of the MOPSO and MOGA algorithms are set by Taguchi method where \( C_1, C_2 \) the number of populations (Pop) and number of generations (Gen) are the parameters of MOPSO and crossover probability and their level values are shown in Table 4. Furthermore, the rest of MOPSO’s parameters are set as \( w_{\text{min}} = 0.4, w_{\text{max}} = 0.9 \) and the time periods \( T_j = 3 \) for \( j = 0, 1, 2, 3 \). The above parameter settings are obtained performing intensive runs. Furthermore, the amount of \( V_{i,j} \) will be obtained automatically after gaining the order quantity \( Q_{i,j} \).
The triangular fuzzy numbers.

Table 3: The general data for a problem with 5 items and 3 periods.

<table>
<thead>
<tr>
<th>Product</th>
<th>$D_{i,1}$</th>
<th>$D_{i,2}$</th>
<th>$\pi_{i,1}$</th>
<th>$\pi_{i,2}$</th>
<th>$\hat{\pi}_{i,1}$</th>
<th>$\hat{\pi}_{i,2}$</th>
<th>$B_i$</th>
<th>$H_i$</th>
<th>$A_i$</th>
<th>$\beta_i$</th>
<th>$S_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1200</td>
<td>800</td>
<td>20</td>
<td>18</td>
<td>10</td>
<td>10</td>
<td>3</td>
<td>5</td>
<td>20</td>
<td>0.5</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>1300</td>
<td>900</td>
<td>20</td>
<td>18</td>
<td>12</td>
<td>12</td>
<td>5</td>
<td>6</td>
<td>25</td>
<td>0.8</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>1500</td>
<td>1200</td>
<td>11</td>
<td>14</td>
<td>8</td>
<td>12</td>
<td>8</td>
<td>6</td>
<td>18</td>
<td>0.8</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>2100</td>
<td>2000</td>
<td>11</td>
<td>14</td>
<td>8</td>
<td>12</td>
<td>8</td>
<td>6</td>
<td>18</td>
<td>0.8</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>1800</td>
<td>1600</td>
<td>12</td>
<td>15</td>
<td>9</td>
<td>11</td>
<td>7</td>
<td>7</td>
<td>19</td>
<td>0.6</td>
<td>6</td>
</tr>
</tbody>
</table>

Table 4: The parameters of the two algorithms and their levels.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Factors</th>
<th>Levels $[1, 2, 3]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOPSO</td>
<td>$C_1$ (A)</td>
<td>$[1.5, 2, 2.5]$</td>
</tr>
<tr>
<td></td>
<td>$C_2$ (B)</td>
<td>$[1.5, 2, 2.5]$</td>
</tr>
<tr>
<td></td>
<td>Pop(C)</td>
<td>$[20, 30, 40]$</td>
</tr>
<tr>
<td></td>
<td>Gen(D)</td>
<td>$[100, 200, 500]$</td>
</tr>
<tr>
<td>MOGA</td>
<td>$P_C$ (A)</td>
<td>$[0.5, 0.6, 0.7]$</td>
</tr>
<tr>
<td></td>
<td>$P_m$ (B)</td>
<td>$[0.08, 0.1, 0.2]$</td>
</tr>
<tr>
<td></td>
<td>Pop(C)</td>
<td>$[30, 40, 50]$</td>
</tr>
<tr>
<td></td>
<td>Gen(D)</td>
<td>$[200, 300, 500]$</td>
</tr>
</tbody>
</table>

The weights associated with the objectives are as triangular fuzzy number $\bar{w} = [\bar{w}_a, \bar{w}_b, \bar{w}_c]$ shown in Figure 4 where membership function of variable $x$ is given by

$$\bar{\mu}(x) = \begin{cases} 0 & x < \bar{w}_a \\ \frac{x - \bar{w}_a}{\bar{w}_b - \bar{w}_a} & \bar{w}_a < x < \bar{w}_b \\ \frac{\bar{w}_c - x}{\bar{w}_c - \bar{w}_b} & \bar{w}_b < x < \bar{w}_c \\ 0 & \bar{w}_c < x. \end{cases} \quad (29)$$

Now, in order to get crisp interval by $\alpha$-cut operation, interval $\bar{w}_{\alpha}$ can be obtained as follows ($\forall \alpha \in [0, 1]$):

$$\frac{\bar{w}_{\alpha}^{(a)} - \bar{w}_{\alpha}}{\bar{w}_b - \bar{w}_a} = \alpha, \quad \frac{\bar{w}_{\alpha} - \bar{w}_{\alpha}^{(a)}}{\bar{w}_c - \bar{w}_b} = \alpha. \quad (30)$$

We have

$$\bar{w}_{\alpha}^{(a)} = (\bar{w}_b - \bar{w}_a) \alpha + \bar{w}_a; \quad \bar{w}_{\alpha}^{(a)} = \bar{w}_c - (\bar{w}_c - \bar{w}_b) \alpha. \quad (31)$$

Therefore,

$$\bar{w}_{\alpha} = \left[\bar{w}_{\alpha}^{(a)}, \bar{w}_{\alpha}^{(c)}\right] = \left[(\bar{w}_b - \bar{w}_a) \alpha + \bar{w}_a, \bar{w}_c - (\bar{w}_c - \bar{w}_b) \alpha\right], \quad (32)$$

where $\bar{w}_1 = [0.3, 0.5, 0.7], \bar{w}_2 = [0.2, 0.3, 0.6]$, and $\alpha = 0.5$.

Table 5: The Taguchi $L_9$ design along with objective values of the algorithms.

<table>
<thead>
<tr>
<th>Run number</th>
<th>$A$</th>
<th>$B$</th>
<th>$C$</th>
<th>$D$</th>
<th>MOPSO</th>
<th>MOGA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>154040</td>
<td>154980</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>154367</td>
<td>154760</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>154220</td>
<td>155075</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>153944</td>
<td>154875</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>153985</td>
<td>155230</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>154568</td>
<td>155102</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>154215</td>
<td>154780</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>154320</td>
<td>154750</td>
</tr>
<tr>
<td>9</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>154100</td>
<td>155111</td>
</tr>
</tbody>
</table>

Table 6: The optimal levels of the algorithms’ parameters for problem 7 of Table 2.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Factors</th>
<th>Optimal levels</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$C_1$</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>$C_2$</td>
<td>2.5</td>
</tr>
<tr>
<td></td>
<td>Pop</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>Gen</td>
<td>200</td>
</tr>
</tbody>
</table>

To perform Taguchi approach in this paper, a $L_9$ design is utilized, based on which the results for problem 7 described in Table 2 are shown in Table 5 as an example. The optimal values of the levels of the algorithms’ parameters shown in Table 5 are represented by Table 6. Figure 5 depicts the mean S/N ratio plot each level of the factors of MOPSO and MOGA for problem 7 in Table 2.

Tables 7 and 8 show the best result obtained by MOPSO and MOGA for the problem with 5 items and 3 periods (problem 7), respectively, including the amounts of decision variables and the optimal objective values. In these tables,
Figure 5: The mean S/N ratio plot for parameter levels of MOPSO and MOGA in problem 7 of Table 2.
Table 7: The best result of the MOPSO algorithm.

<table>
<thead>
<tr>
<th>Product</th>
<th>$Q_{i,1}$</th>
<th>$Q_{i,2}$</th>
<th>$X_{i,2}$</th>
<th>$X_{i,3}$</th>
<th>$V_{i,1}$</th>
<th>$V_{i,2}$</th>
<th>$b_{i,1}$</th>
<th>$b_{i,2}$</th>
<th>TMF</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1215</td>
<td>159</td>
<td>15</td>
<td>0</td>
<td>405</td>
<td>53</td>
<td>0</td>
<td>626</td>
<td>153840</td>
</tr>
<tr>
<td>2</td>
<td>1162</td>
<td>252</td>
<td>0</td>
<td>0</td>
<td>166</td>
<td>36</td>
<td>138</td>
<td>786</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1555</td>
<td>190</td>
<td>55</td>
<td>0</td>
<td>311</td>
<td>38</td>
<td>0</td>
<td>955</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1360</td>
<td>864</td>
<td>0</td>
<td>0</td>
<td>170</td>
<td>108</td>
<td>740</td>
<td>1876</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1435</td>
<td>420</td>
<td>0</td>
<td>0</td>
<td>205</td>
<td>60</td>
<td>365</td>
<td>1545</td>
<td></td>
</tr>
</tbody>
</table>

Table 8: The best result of the MOGA algorithm.

<table>
<thead>
<tr>
<th>Product</th>
<th>$Q_{i,1}$</th>
<th>$Q_{i,2}$</th>
<th>$X_{i,2}$</th>
<th>$X_{i,3}$</th>
<th>$V_{i,1}$</th>
<th>$V_{i,2}$</th>
<th>$b_{i,1}$</th>
<th>$b_{i,2}$</th>
<th>TMF</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1221</td>
<td>168</td>
<td>21</td>
<td>0</td>
<td>407</td>
<td>56</td>
<td>0</td>
<td>611</td>
<td>154550</td>
</tr>
<tr>
<td>2</td>
<td>959</td>
<td>392</td>
<td>0</td>
<td>0</td>
<td>137</td>
<td>56</td>
<td>341</td>
<td>849</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1220</td>
<td>390</td>
<td>0</td>
<td>0</td>
<td>244</td>
<td>78</td>
<td>280</td>
<td>1090</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1168</td>
<td>960</td>
<td>0</td>
<td>0</td>
<td>146</td>
<td>120</td>
<td>932</td>
<td>1972</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>168</td>
<td>2254</td>
<td>0</td>
<td>0</td>
<td>24</td>
<td>322</td>
<td>1632</td>
<td>978</td>
<td></td>
</tr>
</tbody>
</table>

Table 9: The ANOVA analysis of the performances.

<table>
<thead>
<tr>
<th>Source</th>
<th>DF</th>
<th>SS</th>
<th>MS</th>
<th>$F$</th>
<th>$P$ value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Factor</td>
<td>1</td>
<td>$1.11E + 11$</td>
<td>$1.11E + 11$</td>
<td>0.28</td>
<td>0.6</td>
</tr>
<tr>
<td>Error</td>
<td>78</td>
<td>$3.11E + 13$</td>
<td>$3.99E + 11$</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Total</td>
<td>79</td>
<td>$3.12E + 13$</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

TMF is the best value of the biobjective inventory planning problem, which is given in the last two columns of Table 2. Similarly, the best TMF for the other problems is obtained and is summarized in Table 2.

To compare the performances of the MOPSO and MOGA, several statistical and graphical approaches are employed. A one-way ANOVA analysis of the means of the algorithms in confidence 0.95% is used to compare and evaluate the objective values of the generated 40 problems. Table 9 shows the ANOVA analysis of the results of the two algorithms that demonstrates no significant difference between both algorithms. Moreover, the mean and standard deviation (Std. Dev) of the objective values of the 30 generated problems shows that the MOPSO has the better performance in terms of the objective values in comparison with the MOGA. In addition, a pictorial presentation of the performances of the two algorithms shown by Figure 6 displays that the MOPSO is more efficient than the MOGA algorithm in the large number of the problems.

Figure 7 depicts the boxplot and the individual value plot and Figure 8 shows the residual plots for the algorithms.

A comparison of the results in Table 2 shows that the MOPSO algorithm performs better than the MOGA in terms of the fitness functions values.

6. Conclusion and Recommendations for Future Research

In this paper, a biobjective multi-item multiperiod inventory planning problem with total available budget under all unit discount for some items and incremental quantity discount for other items was considered. The orders were assumed to be placed in batch sizes and the order quantities at the end period were zeros. Shortages were allowed and contain backorder and lost sale. It was assumed that the beginning inventory level in primary period was zeros and the order quantity in each period was more than the shortage quantity in the previous period. Due to adopting decisions related to a certain department of production planning (extending warehouse or building a new manufacturing line), the manager decided to build a new warehouse for the ordered items. The objectives were to minimize both the total inventory costs...
and the total required storage space, for which a weighted combination was defined as the objective function. The aim of the study was to determine the optimal order quantity and the shortage quantity of each product in each period such that the objective function is minimized and the constraints hold. The developed model of the problem was shown to be an integer nonlinear programming mixed with binary variables. To solve the model, both a multiobjective particle swarm optimization and a multiobjective genetic algorithm were applied. The results showed that for the 10 specific problems the MOPSO performs better than the MOGA in terms of the fitness function values.

Some recommendations for future works are to expand the model to cover a supply chain environment, to consider fuzzy or stochastic demands, and/or to take into account the inflation and the time value of the money.

**Conflict of Interests**

The authors declare that there is no conflict of interests regarding the publication of this paper.
Acknowledgments

The authors would like to acknowledge the University of Malaya for providing the necessary facilities and resources for this research. This research was fully funded by the Ministry of Higher Education, Malaysia, with the high impact research (HIR) Grant no. HIR-MOHE-16001-D00001.

References


Submit your manuscripts at http://www.hindawi.com